MODUL PRAKTIKUM VIII
INTERFACE* DAN *ABSTRACT

A. Tujuan

Mahasiswa mampu menerapkan dan menganalisa penggunaan *interface* dan *abstract* dalam Java

B. Latihan

Latihan 1:

Implementasikan *source code* berikut :

1. Hewan.java

```
abstract class Hewan {
    String nama;
    public abstract void habitatHewan();
    public void namaHewan() {
        System.out.println("\nMethod di dalam abstract class Hewan");
        System.out.println("Nama hewan    : " + nama);
    }
}
```

2. Karnivora.java

```
class Karnivora extends Hewan{
    String habitat;
    public void habitatHewan()
    {
        System.out.println("\nMethod di dalam class Karnivora");
        System.out.println("Habitat hewan : "+habitat);
    }
}
```

3. TesHewan.java

```
public class TesHewan {
    public static void main(String[] args) {
        Karnivora singa = new Karnivora();
        singa.nama = "Singa";
        singa.habitat = "Darat";
        singa.namaHewan();
        singa.habitatHewan();
    }

}
```

Output dari program di atas adalah:

Latihan 2:

Implementasikan *source code* berikut :

1. Relation.java

```
public interface Relation {
    public boolean isGreater(Object a, Object b);
    public boolean isLess(Object a, Object b);
    public boolean isEqual(Object a, Object b);
}
```

2. Line.java

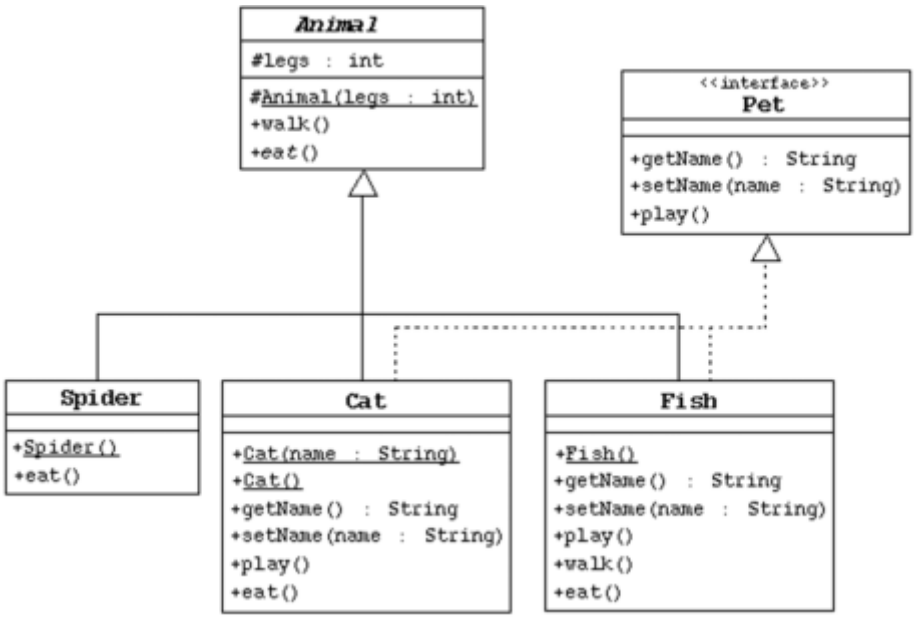
```
public class Line implements Relation{
    private double x1,x2,y1,y2;

    public Line(double x1, double x2, double y1, double y2){
        this.x1 = x1;
        this.x2 = x2;
        this.y1 = y1;
        this.y2 = y2;
    }

    public double getLength(){
        double length = Math.sqrt((x1-x2)*(x1-x2)+(y1-y2)*(y1-y2));
        return length;
    }
    public boolean isGreater(Object a, Object b){
        double aLen = ((Line)a).getLength();
        double bLen = ((Line)b).getLength();
        return (aLen >bLen);
    }

    public boolean isLess(Object a, Object b){
        double aLen = ((Line)a).getLength();
        double bLen = ((Line)b).getLength();
        return (aLen <bLen);
    }

    public boolean isEqual(Object a, Object b){
        double aLen = ((Line)a).getLength();
        double bLen = ((Line)b).getLength();
        return (aLen == bLen);
    }
}
```

a. Tambahkan sebuah *main class* untuk menguji program di atas

b. Output dari program di atas adalah:

C. Tugas Praktikum

Implementasikan UML *class diagram* berikut dalam program Java



1. *Class* Animal.java

- Animal adalah *abstract superclass* dari semua hewan.
- Buatlah *protected integer attribute* dengan nama `legs`, atribut ini digunakan untuk menyimpan informasi jumlah kaki hewan.
- Buatlah *protected constructor* yang digunakan untuk menginisialisasi variabel `legs`.
- Buatlah *abstract method* `ea()`.
- Buatlah *concrete method* `walk()` yang digunakan untuk menampilkan tulisan tentang bagaimana hewan berjalan dan jumlah kaki hewan tersebut
  (misal: hewan ini berjalan dengan 4 kaki).

2. *Class* Spider.java

- *Class* Spider merupakan anak dari *class* Animal.
- Buatlah *constructor* yang digunakan untuk memanggil *superclass constructor*, *constructor* ini juga digunakan untuk menginisialisasi jumlah kaki Spider (kita tahu bahwa semua Spider pasti mempunyai kaki sebanyak 8 buah.
- Implementasikan *method* `eat()`.

3. *Interface* Pet.java
   Buatlah *interface* Pet sesuai dengan UML *class diagram*.

4. *Class* Cat.java

- *Class* Cat adalah anak dari *class* Animal dan mengimplementasikan interface Pet.
- Buatlah variabel `name` yang bertipe String yang digunakan untuk menyimpan nama Cat. (variabel ini tidak digambarkan pada UML diagram).

- Buatlah *constructor* dengan satu argumen bertipe String yang digunakan untuk mengeset nama Cat. *Constructor* ini juga harus memanggil *superclass constructor* untuk mendefinisikan bahwa Cat mempunyai kaki sebanyak 4 buah.
- Buatlah *constructor* lain yang tidak mempunyai argumen. Buat *constructor* ini supaya memanggil *constructor* pada poin sebelumnya (dengan menggunakan kata kunci *this*) dan passing *empty string* sebagai argumen (*empty string* / " ").
- Implementasikan *method-method* yang ada pada *interface* Pet.
- Implementasikan *method* `eat()`.

5. *Class* Fish.java

- Class Fish adalah anak dari *class* Animal.
- Lakukan *override* pada semua *method* Animal dan definisikan bahwa ikan tidak berjalan tetapi berenang.
- Class Fish mengimplementasikan *interface* Pet.
- Jangan lupa untuk mendefinisikan *method* kepunyaan *interface*.

6. *Class* TestAnimals.java

```
public class TestAnimals {
  public static void main(String[] args) {
      Fish f = new Fish();
      Cat c = new Cat("Fluffy");
      Animal a = new Fish();
      Animal e = new Spider();
      Pet p = new Cat();

      // Demonstrate different implementations of an interface
      f.play();
      c.play();

      // Demonstract virtual method invocation
      e.eat();
      e.walk();

      // Demonstrate calling super methods
      a.walk();
  }
 }
```